A Sierra Monitor Company

Driver Manual
(Supplement to the FieldServer Instruction Manual)

FS-8700-52 Notifier AM6000

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after May 1, 2001

Instruction Manual Part Number FS-8700-52

5/22/2002

TABLE OF CONTENTS

1. Notifier AM6000 Description

The Notifier AM6000 driver is a seial driver. It allows the FieldServer to transfer data from a Notifier AM6000 panel over either RS232 or RS485 using Notifier AM6000 protocol. There are eight RS232 and two RS485 ports standard on the FieldServer. The FieldServer can only a passive Client.

The driver receives messages intended for a system printer, interprets these messages by filling in data arrays in the FieldServer. This data is available for other devices or PLC's to read.

2. Driver Scope of Supply

2.1 Supplied by FieldServer for this driver

FieldServer Technologies PART #	DESCRIPTION
8915-10	UTP cable (7 foot) for RS232 use
	UTP cable (7 foot) for Ethernet connection
8917-02	RJ45 to DB9F Connector adapter
8917-01	RJ45 to DB25M connection adapter
SPA59132	RS485 connection adapter
	Driver Manual.

3. Hardware Connections

Configure the Notifier AM6000 according to manufacturer's instructions.

4. Configuring the FieldServer as a Notifier AM6000 Client

For a detailed discussion on FieldServer configuration, please refer to the instruction manual for the FieldServer. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files on the driver diskette).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a Notifier AM6000 Server.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Notifier AM6000 communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

4.1 Data Arrays

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Format	Provide data format. Each data array can only take on one format.	FLOAT, BIT, UInt16, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required for the data being placed in this array.	1-10,000

Example

```
//    Data Arrays
//
Data_Arrays
Data_Array_Name,     Data_Format,     Data_Array_Length
DA_AI_01,            UInt16,               200
DA_AO_01,            UInt16,               200
DA_DI_01,            Bit,                  200
DA_DO_01,            Bit,                  200
```

4.2 Client Side Connections

Section Title
Connections

Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2
Baud*	Specify baud rate	2400 (Only baud rate supported by the Notifier port)
Parity*	Specify parity	Even
Data_Bits*	Specify data bits	7
Stop_Bits*	Specify stop bits	**1**
Protocol	Specify protocol used	AM6000
Handshaking*	Specify hardware handshaking	**None**
Poll Delay*	Time between internal polls	0-32000 seconds **default 1 second**

Example

```
//    Client Side Connections

Connections
Port, Baud, Parity, Protocol, Data_bits, Handshaking, Poll_Delay
P8,   2400, Even,   AM6000,   7,         None,        0.100s
```

4.3 Client Side Nodes

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Modbus station address of physical server node	1-255
Protocol	Specify protocol used	Modbus_RTU, Modbus/TCP, etc.
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2

Example

```
//    Client Side Nodes

Nodes
Node_Name, Protocol,   Port
Panel1,    AM6000,     P8
```

4.4 Client Side Map Descriptors

4.4.1 FieldSe rver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Location	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor	Passive

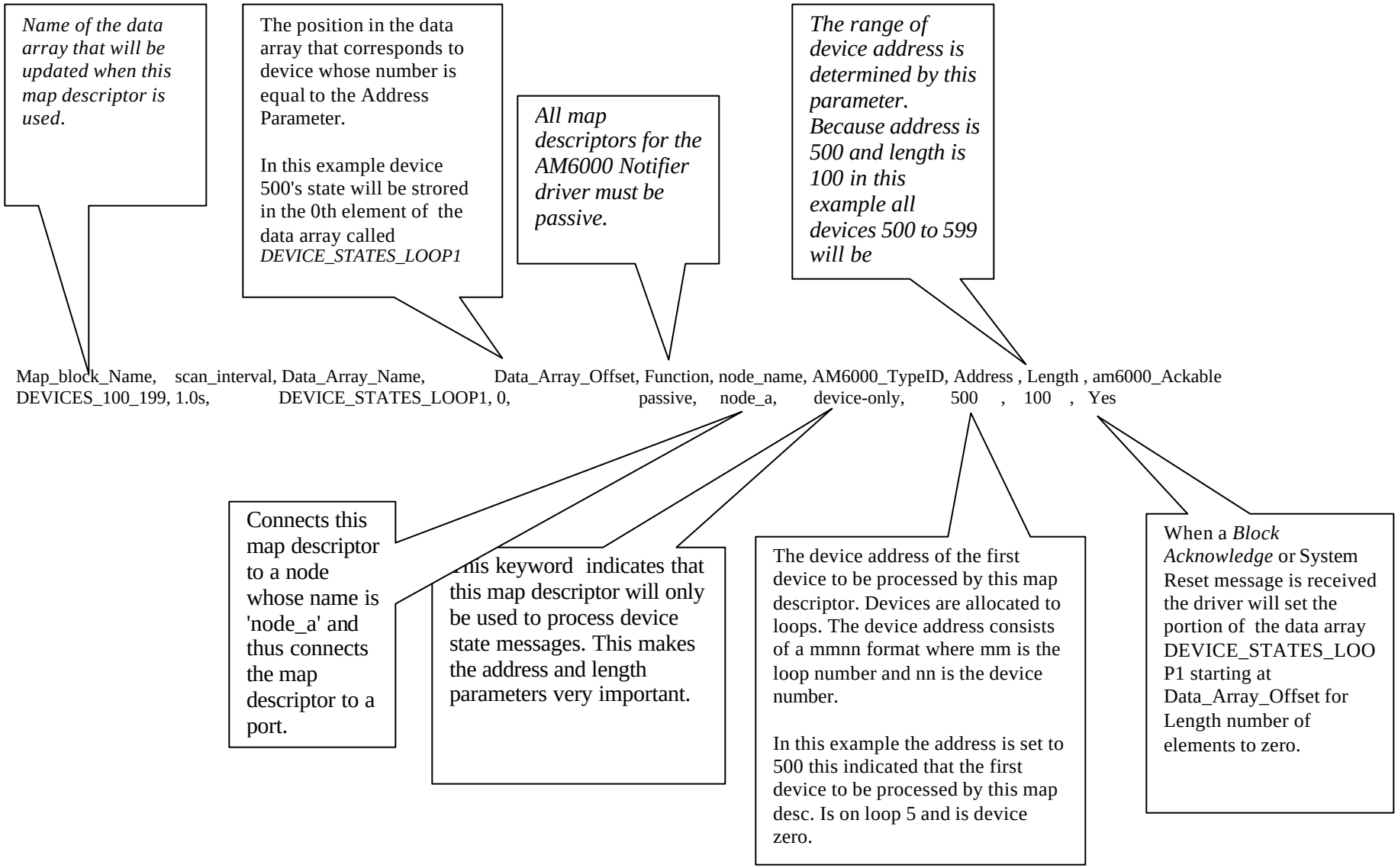
4.4.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
Data_Type	Data type	Register, Coil, AI, DI
Length	Length of Map Descriptor. This value is important when a 'Block Ack' message is received as it tells the driver how much of the data array to zeroise.	1 - 1000
Address	Loop/Device Address In the format mmnn where mm is the loop number and nn is the device number. Together they forma a composite device number.	501, 302
AM6000 Driver specific parameters & keywords.		
Am6000_typeID	Used to differentiate between device and system type messages.	Device-only, System_trbl, device&type. system-Styl2, catch-all Additional notes are provided in section 5.1
Am6000_Status	Allows direction of alarm or trouble states to different data arrays.	**Any**, Alarm, trouble, erase, on , eraseon Additional notes are provided in section 5.2
AM6000_Ackable	Tells the driver that the data area of this mapdesc must be Zeroised when a block ack is received.	Additional notes are provided in section 5.3
Am6000_simulation	Provided for debugging and test purposes only. Tells the driver which debug message to send. This parameter is for use by FieldServer Technologies only.	

4.4.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	>0.1s

4.4.4 Map Descriptor Example.



Name of the data array that will be updated when this map descriptor is used.

All map descriptors for the AM6000 Notifier driver must be passive.

When a Block Acknowledge or System Reset message is received the driver will set the portion of the data array DEVICE_STATES_LOOP1 starting at Data_Array_Offset for Length number of elements to zero.

```
Map_block_Name,scan_interval, Data_Array_Name,      Data_Array_Offset, Function, node_name, AM6000_TypeID, Length , am6000_Ackabl
MD1,              1.0s,          SYSTEM_TROUBLES2, 5,                 passive,    node_a,    system-styl2,    50     ,  Yes
```

System Trouble Messages are processed using this map descriptor. Only one map descriptor like this is necessary for a whole system.

This indicates Style2 is being used.

Thus when system trouble message number ***n*** is processed the nth element of data array starting at location 5 (this example) will be set to 1.

5. Driver Notes

5.1 Map Descriptor Keywords

5.1.1 AM6000_TypeID

This keyword sets up a category of messages. If you are interested in device state's use the keyword ***device-only***. If you are interested in system trouble messages use the keyword ***system-trbl*** (An alternate method for reporting system trouble messages is provided using the keyword ***system-styl2***.)

device-only

Only device based messages are processed using map descriptors with this keyword. The ***Address*** and ***Length*** parameters define the range of devices processed using the map descriptor. A device address consists of a loop and device number. This driver treats the two parts as a single device address. Thus loop 5 device 2 becomes address 502 for this driver and a map descriptor with an address of 500 and a length of 100 would process a message from device 2 on loop 5.

system-trbl

System Trouble messages are processed a numeric value is stored in the first element of the map descriptor's data array. If a new system trouble message is processed then the value is overwritten with the new value.

system-styl2

This is an alternate style for processing *System Trouble* messages. If system trouble message whose index in the list below is n is received then the nth element of the data array is set to one. Make sure the length parameter is set to at least 50.

Catch-all

The use of this keyword is described in section 5.2

5.1.2 AM6000_Status

Device state messages report a number of different states for the device. These states are refered to as the device status.

The following are possible values of this parameter. Any, alarm, trouble, on, erase, eraseon.

If you do not use this parameter in a ***device-only*** map descriptor then the driver uses the default value of ***Any***. This means that any device state message will result in the same data array being updated. Thus a trouble/on/alarm message will result in the array being set to a 1.

If you want to maintain separate array's for each state the you use this parameter. For one device address range you would have multiple map descriptors, each with a different AM6000_Status keyword.

You should note that the keyword, ***erase***, results in the data array value being set to zero when a message reports the device state as 'ERASE'. The keyword ***eraseon*** sets the value to one, when the same message is received.

5.1.3 AM6000_Ackable

This parameter tells the driver that the data array portion associated with this map descriptor can be set to zero when a Block Acknowledge or System Reset message is received.

You set this parameter to one of the following legal keywords: ***yes, no.***

The Data_Array_Location and Length parameters are used to determine what portion of the associated data array must be set to zero.

5.1.4 AM6000_Simulation

This keyword is for used by FieldServer Technologies Engineers and is used for testing this driver.

5.2 Statistics

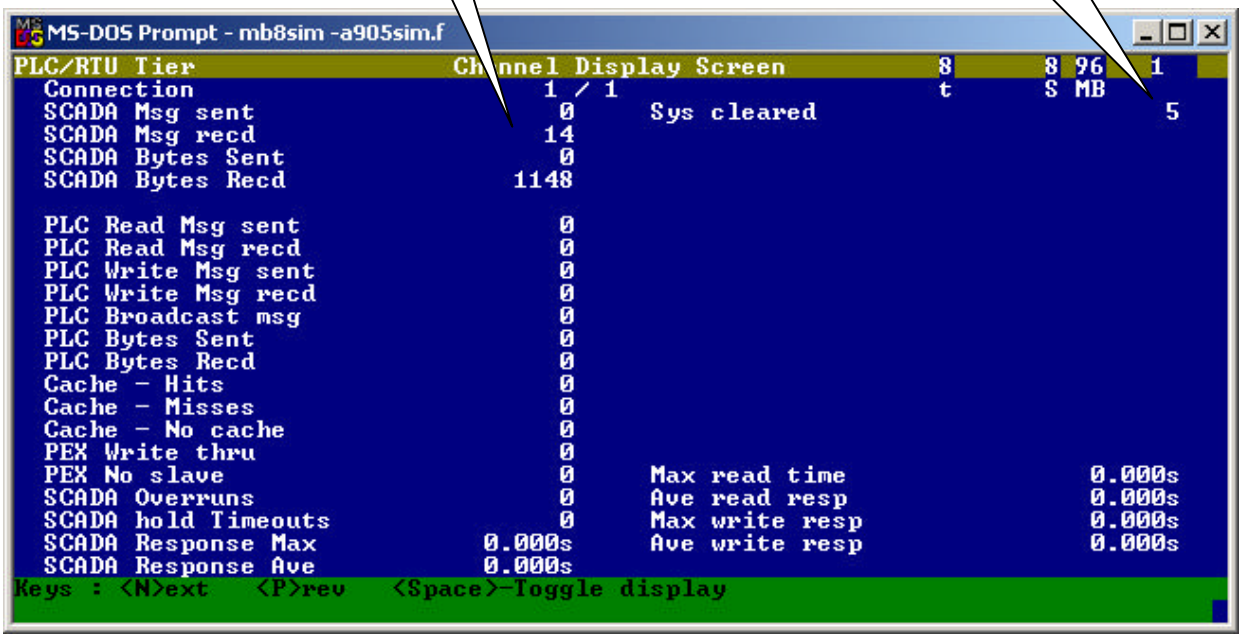
This driver does not keep statistics for each map descriptor. Statistics are maintained for the connection to the Am6000 Notifier device.

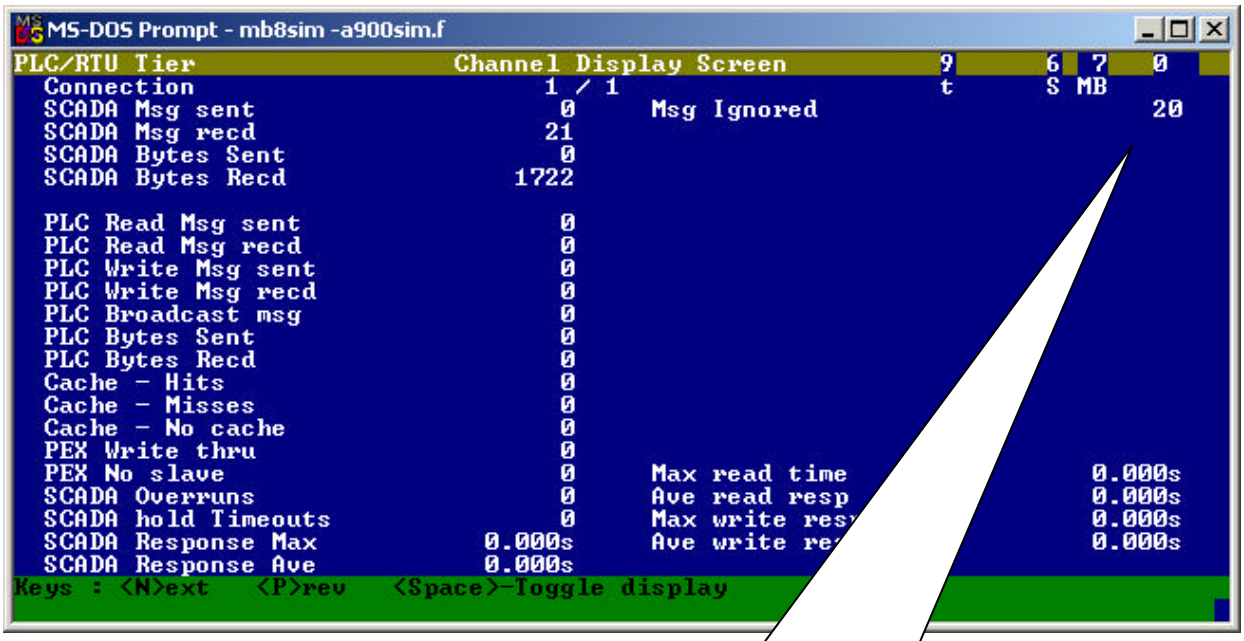
Count of received messages and bytes. A complete received message is 82 bytes long and thus if all messages are received correctly the byte count should be a multiple of 82.

This indicates the number of times a 'Block Ack' or 'System Reset' message have been applied.

If you have 5 'Ackable' map descriptors and one 'Block Ack' message was received a count of 5 would be reported. (One message applied 5 times).

You will need to count the number of 'ackable' map descriptors to use this statistic meaningfully.





This statistic indicates the number of messages that were received but were ignored by the driver. Messages are ignored for one of two reasons. Firstly, the driver might not understand the message and secondly because the driver doesn't know what to do with the data from the message.

Ignored Messages

Ignored messages are very important since they do not result in the data arrays being updated. Messages are ignored for one of two reasons.

Firstly, the driver might not understand the message and secondly because the driver doesn't know what to do with the data from the message. The message t may contain a keyword or be formatted in a way that cannot be understood.. It may be the case that the equipment manufacturer adds new features to the protocol. FieldServer Technologies needs to be informed of any such messages so that this driver can be updated.

Secondly, you may have omitted to define a map descriptor which tells the driver what to do with the data from an incoming message. For example, say a map descriptor is defined for address 501 with a length of 50 but a message is received from device 575. Clearly the map descriptor's device address range does not extend ass far as 575 and thus the driver doesn't know where to store device 575's data.

Catching Ignored Messages

To assist you catch and monitor ignored messages the driver provides a special map descriptor keyword.

Make a map descriptor and set the parameter ***AM6000_TypeID***'s value to ***Catch-All***. Make sure that this is the last map descriptor in the csv file. The map descriptor requires a data length of at least 82 and when using RUI_Debug you should view the data array in <S>tring mode.

The driver will place any ignored messages in this buffer. You will be able to read the message in <S>tring mode an make a decision on the necessary corrective action. If there are multiple messages being ignored the buffer will be overwritten.

You could use your PLC / control device / Scada to monitor the first byte of this data array and generate an alarm if the value is non-zero. Thus, even though a message has been ignored you system will know about it.

5.3 Listing of System Trouble Messages

Msg Index	Message
0	!!! ILLEGAL !!!
1	MAINS TROUBLE
2	POWER LOW ON MAIN SUPPLY
3	POWER LOW ON AUXILIARY SUPPLY
4	POWER LOW ON BATTERY-CHARGER
5	POWER OVERVOLTAGE ON MAIN SUPPLY
6	POWER OVERVOLTAGE ON BATTERY-CHARGER
7	FAILURE OR OVERLOAD ON MAIN SUPPLY
8	FAILURE OR OVERLOAD ON AUXIL. SUPPLY
9	FAILURE ON BATTERY-CHARGER
10	DISCONNEC. BATT. OR FUSE FAILURE
11	BATTERY EXHAUSTED
12	BATTERY CHARGER UNBALANCED
13	AUX. SUPPLY DISCONNECTED
14	AUX. SUPPLY USER FUSE
15	MAIN SUPPLY USER FUSE
16	SIREN FUSE
17	EARTH TROUBLE
18	SIREN SUPERV.LINE INTERRUPTED
19	BATTERY FAILURE
20	WIRE CUT ON LINE
21	SHORT CIRCUIT ON LINE
22	PRINTER: OFF LINE/BUFFER FULL/PAPER END
23	COMMUNICATION ERROR ON
24	LOW VOLTAGE ON LINE
25	SIDE A OPEN ON LINE
26	SIDE B OPEN ON LINE
27	SYSTEM KEYPAD TROUBLE
28	INTERNAL PROGR.ENABLING JUMPER :ENABLED
29	SYSTEM START UP
30	CPU RESET OR WATCH-DOG FAILURE
31	CRT-TERMINAL : OFF-LINE
32	FLASH MEMORY ERROR ON
33	SIREN SUPERV.LINE
34	LINE
35	ANNUNC.